

1  
2 **CLAIMS**

3       1.     A method for testing software comprising:  
4       modeling software using a software model that describes behavior  
5       associated with the software; and  
6       operating on the software model using a random destination algorithm and  
7       at least one other different algorithm to produce a sequence of test actions, the  
8       random destination algorithm being configured to randomly select a destination in  
9       the model and move to that destination to produce the sequence of test actions.

10  
11       2.     The method of claim 1, wherein the software model comprises a state  
12       graph having multiple nodes individual ones of which representing a state, and  
13       links between the nodes that represent actions.

14  
15       3.     The method of claim 2, wherein said operating comprises using the  
16       random destination algorithm to select a destination node at random, independent  
17       of a present node, and traverse state space to arrive at the destination node.

18  
19       4.     The method of claim 2, wherein said operating comprises using the  
20       random destination algorithm to select a destination node at random, independent  
21       of any previously-traversed nodes, and traverse state space to arrive at the  
22       destination node.

1           5.     The method of claim 2, wherein said operating comprises using the  
2 random destination algorithm to select a destination node at random, independent  
3 of a nearest neighbor node, and traverse state space to arrive at the destination  
4 node.

5  
6           6.     The method of claim 2, wherein the software model comprises  
7 clusters of related nodes, and said operating comprises using the random  
8 destination algorithm to select, at random, at least one cluster of nodes.

9  
10          7.     The method of claim 2, wherein the software model comprises  
11 clusters of related nodes, and said operating comprises using the random  
12 destination algorithm to select, at random, at least one node inside at least one  
13 cluster of nodes.

14  
15          8.     One or more computer-readable media having computer-readable  
16 instructions thereon which, when executed by a computer, cause the computer to:

17               model software using a software model that describes behavior associated  
18 with the software, the software model comprising a state graph having multiple  
19 nodes individual ones of which represent a state, and links between the nodes that  
20 represent actions; and

21               operate on the software model using a random destination algorithm and at  
22 least one other different algorithm to produce a sequence of test actions, the  
23 random destination algorithm being configured to randomly select a destination  
24 node in the model and move to that destination node to produce the sequence of  
25

test actions, the selection of the destination node being performed independent of any previously-traversed nodes, and independent of any nearest neighbor nodes.

**9.** A method of testing software comprising:

modeling software using a software model that describes behavior associated with the software;

operating on the software model using a random destination algorithm to produce a sequence of test actions, the random destination algorithm being configured to randomly select a destination in the model and move to that destination to produce the sequence of test actions; and

operating on the software model using multiple other algorithms that are different from the random destination algorithm to produce a further sequence of test actions.

**10.** The method of claim 9, wherein said modeling comprises using a state graph having multiple nodes individual ones of which represent a state, and links between the nodes that represent actions.

**11.** The method of claim 9, wherein said multiple other algorithms comprise a random walk algorithm.

**12.** The method of claim 9, wherein said multiple other algorithms comprise a Chinese postman algorithm.

1           **13.**    The method of claim 9, wherein said multiple other algorithms  
2 comprise a Markov chain algorithm.

3  
4           **14.**    The method of claim 9, wherein said multiple other algorithms  
5 comprise a anti-random walk algorithm.

6  
7           **15.**    The method of claim 9, wherein said multiple other algorithms  
8 comprise an algorithm selected from a group comprising: a random walk  
9 algorithm, a Chinese postman algorithm, a Markov chain algorithm, and a anti-  
10 random walk algorithm.

11  
12          **16.**    One or more computer-readable media having computer-readable  
13 instructions thereon which, when executed by a computer, cause the computer to:

14           operate on a software model using a random destination algorithm to  
15 produce a sequence of test actions, the software model comprising a state graph  
16 having multiple nodes individual ones of which represent a state, and links  
17 between the nodes that represent actions, the random destination algorithm being  
18 configured to randomly select a destination node in the state graph and move to  
19 that destination node to produce the sequence of test actions; and

20           operate on the software model using multiple other algorithms that are  
21 different from the random destination algorithm to produce a further sequence of  
22 test actions, the multiple other algorithms being selected from a group comprising:  
23 a random walk algorithm, a Chinese postman algorithm, a Markov chain  
24 algorithm, and a anti-random walk algorithm.

1        **17.**    A method of testing software comprising:

2        traversing a state graph that models software, the state graph having  
3        multiple nodes individual ones of which represent a state, and links between the  
4        nodes that represent actions, said traversing using an algorithm having a first  
5        graph traversal characteristic to produce a sequence of test actions; and

6        traversing the state graph using an algorithm having a second graph  
7        traversal characteristic that is different from the first graph traversal characteristic  
8        to produce a further sequence of test actions.

9  
10       **18.**    The method of claim 17, wherein the algorithms are different.

11  
12       **19.**    The method of claim 17, wherein the algorithm having the first  
13       graph traversal characteristic is one selected from a group of algorithms  
14       comprising: a random walk algorithm, a random destination algorithm, and a anti-  
15       random walk algorithm.

16  
17       **20.**    The method of claim 19, wherein the algorithm having the second  
18       graph traversal characteristic different from the algorithm having the first graph  
19       traversal characteristic and is one selected from a group of algorithms comprising:  
20       a random walk algorithm, a random destination algorithm, and a anti-random walk  
21       algorithm.

1           **21.**   One or more computer-readable media having computer-readable  
2 instructions thereon which, when executed by a computer, cause the computer to  
3 implement the method of claim 17.

4  
5           **22.**   A method of testing software comprising:  
6           traversing a state graph using a deterministic first algorithm to produce a  
7 sequence of test actions, the state graph having multiple nodes individual ones of  
8 which represent a state, and links between the nodes that represent actions; and  
9           traversing the state graph using a second algorithm that is less deterministic  
10 than the first algorithm to produce a further sequence of test actions.

11  
12          **23.**   A method of testing software comprising:  
13          traversing a state graph using a random walk first algorithm to produce a  
14 sequence of test actions, the state graph having multiple nodes individual ones of  
15 which represent a state, and links between the nodes that represent actions; and  
16          traversing the state graph using a second algorithm that is less random than  
17 the first algorithm to produce a further sequence of test actions.

18  
19          **24.**   A method of testing software comprising:  
20          providing one or more algorithms for operating on a software model that  
21 describes behavior associated with software that is to be tested;  
22          selecting one or more algorithms;  
23          operating on the software model using the selected one or more algorithms  
24 to produce a sequence of test actions;  
25          changing the selected one or more algorithms; and

1 operating on the software model using one or more changed algorithms.

2  
3 **25.** The method of claim 24, wherein said changing comprises changing  
4 a way an algorithm interacts with the software model.

5  
6 **26.** The method of claim 25, wherein said changing comprises changing  
7 one or more properties associated with an algorithm.

8  
9 **27.** The method of claim 24, wherein said changing comprises selecting  
10 at least one different algorithm.

11  
12 **28.** One or more computer-readable media having computer-readable  
13 instructions thereon which, when executed by a computer, cause the computer to:

14 provide one or more algorithms for operating on a software model that  
15 describes behavior associated with software that is to be tested;

16 select multiple algorithms to define a first collection of algorithms;

17 operate on the software model using the first collection of algorithms to  
18 produce a sequence of test actions;

19 change at least one of the selected algorithms to define a second collection  
20 of algorithms; and

21 operate on the software model using the second collection of algorithms to  
22 produce an additional sequence of test actions.

1           **29.**    A method of testing software comprising:  
2           traversing a state graph using a random destination algorithm, the state  
3           graph having multiple nodes individual ones of which representing a state, and  
4           links between the nodes that represent actions, said traversing producing a  
5           sequence of test actions; and

6           traversing the state graph using multiple steps from a random walk  
7           algorithm to produce an additional sequence of test actions.  
8

9           **30.**    The method of claim 29 further comprising traversing the state  
10          graph using a random destination algorithm after said traversing of the state graph  
11          using the random walk algorithm.  
12

13          **31.**    The method of claim 29, wherein said traversing using multiple  
14          steps comprises using a predetermined number of steps.  
15

16          **32.**    The method of claim 29, wherein said traversing using multiple  
17          steps comprises using a random number of steps.  
18

19          **33.**    The method of claim 29, wherein said acts of traversing comprise  
20          iterating through the random destination and random walk algorithms.  
21

22          **34.**    The method of claim 33, wherein said traversing using multiple  
23          steps comprises changing the number of steps on at least one iteration.  
24  
25



1           **35.**    The method of claim 33, wherein said traversing using multiple  
2 steps comprises randomly changing the number of steps on at least one iteration.

3  
4           **36.**    The method of claim 33, wherein said traversing using multiple  
5 steps comprises changing the number of steps on at least one iteration in  
6 accordance with probabilistic characteristics.

7  
8           **37.**    A method of testing software comprising:  
9           selecting a first algorithm from among a number of different algorithms;  
10          operating on a software model that describes behavior of software that is to  
11 be tested, said operating taking N steps using the first algorithm, where N is an  
12 integer and said steps produce a sequence of test actions;  
13          selecting a second algorithm from among the number of different  
14 algorithms, the second algorithm being different from the first algorithm; and  
15          operating on the software model by taking N1 steps using the second  
16 algorithm, where N1 is an integer, said N1 steps producing an additional sequence  
17 of test actions.

18  
19          **38.**    The method of claim 37, wherein the algorithms are different based  
20 upon how they interact with the software model.

21  
22          **39.**    The method of claim 37, wherein the algorithms are different based  
23 upon how they traverse a graph that describes the software's behavior.

1        40.    The method of claim 37, wherein said number of different  
2 algorithms include at least one algorithm that is more deterministic than at least  
3 one other algorithm.

4  
5        41.    The method of claim 37, wherein said number of different  
6 algorithms include at least one algorithm that is more random than at least one  
7 other algorithm.

8  
9        42.    The method of claim 37, wherein at least one of said acts of  
10 selecting is based on the structure of the software model.

11  
12       43.    The method of claim 37, wherein N and N1 are predetermined.

13  
14       44.    The method of claim 37, wherein N and N1 are preprogrammed.

15  
16       45.    The method of claim 37, wherein N and N1 are randomly selected.

17  
18       46.    The method of claim 37, wherein N and N1 are calculated using a  
19 Poisson distribution having multiple values each with an assigned probability of  
20 being selected.

21  
22       47.    The method of claim 46, wherein one or more assigned probabilities  
23 change over time.

1           **48.**    The method of claim 37 further comprising iterating through said  
2 acts of operating at least one time.

3  
4           **49.**    The method of claim 48 further comprising changing the values of  
5 N and N1 during the iteration.

6  
7           **50.**    The method of claim 48 further comprising:  
8 assigning values to N and N1 using a first method on a first pass; and  
9 assigning values to N and N1 using a second method that is different from  
10 the first method on a second pass.

11  
12          **51.**    The method of claim 37 further comprising replacing one or more of  
13 the algorithms used to operate on the software model after a certain period of time.

14  
15          **52.**    The method of claim 37 further comprising replacing one or more of  
16 the algorithms used to operate on the software model after the one or more  
17 algorithms have been used a certain number of times.

18  
19          **53.**    A method of testing software comprising:  
20 representing software using a model that describes the software's behavior,  
21 the software having an associated social context; and  
22 selecting one or more algorithms to operate upon the model as a function of  
23 the software's social context; and  
24 operating upon the model using the selected one or more algorithms to  
25 produce a sequence of test actions.

1  
2       **54.**    The method of claim 53, wherein the social context is associated  
3 with a software developer who developed the software.  
4

5       **55.**    The method of claim 53 further comprising:  
6       changing the one or more algorithms; and  
7       operating upon the model using changed algorithms to produce an  
8 additional sequence of test actions.  
9

10       **56.**    A method of testing software comprising:  
11       developing a profile associated with one or more software developers, the  
12 profile describing one or more algorithms that are more likely to identify problems  
13 associated with software developed by the one or more software developers;  
14       selecting, from a developer's profile, one or more algorithms when a  
15 software model associated with the developer's software is to be operated upon;  
16 and  
17       operating upon the software model using the selected one or more  
18 algorithms to produce a sequence of test actions.  
19

20       **57.**    A method of testing software comprising:  
21       defining one or more clusters in a software model that models software that  
22 is to be tested;  
23       providing multiple different algorithms for operating upon the software  
24 model;  
25

1 selecting a first algorithm for operating on the software model to produce a  
2 sequence of test actions;

3 selecting a second algorithm that is different from the first algorithm for  
4 operating on the software model to produce an additional sequence of test actions;  
5 and

6 operating on the software model using the first and second algorithms to  
7 produce the sequences of test actions, one of the first and second algorithms  
8 having a better chance at accessing a cluster than the other of the first and second  
9 algorithms.

10  
11 **58.** The method of claim 57, wherein said software model comprises a  
12 state graph having multiple nodes and links between the nodes, individual nodes  
13 representing states, individual links representing actions that move between states.

14  
15 **59.** The method of claim 58, wherein the first and second algorithms  
16 have different graph traversal characteristics.

17  
18 **60.** The method of claim 58, wherein said defining comprises defining  
19 the clusters based on areas of connectivity within the state graph.

20  
21 **61.** The method of claim 57, wherein said defining comprises defining  
22 the clusters based on the structure of the software.

1           **62.**    The method of claim 57, wherein one of said first and second  
2 algorithms comprises a random destination algorithm.

3  
4           **63.**    The method of claim 57, wherein one of said first and second  
5 algorithms comprises a random walk algorithm.

6  
7           **64.**    The method of claim 57, wherein one of said first and second  
8 algorithms comprises a anti-random walk algorithm.

9  
10          **65.**    The method of claim 57, wherein one of said first and second  
11 algorithms comprises a deterministic algorithm, and the other algorithm comprises  
12 a non-deterministic algorithm.

13  
14          **66.**    A software-testing system comprising:

15 a software model processor configured to:

16                receive a software model that describes behavior associated with  
17 software that is to be tested, and

18                operate upon the model to provide a sequence of test commands for  
19 testing the software; and

20                an algorithm set associated with the model processor and comprising  
21 multiple different algorithms, the software model processor being configured to  
22 select at least two different algorithms and use the algorithms to operate upon the  
23 software model to produce the sequence of test commands.

1           **67.**   The software-testing system of claim 66, wherein the model  
2 processor is configured to change one or more of the algorithms.

3  
4           **68.**   The software-testing system of claim 66, wherein at least one  
5 algorithm comprises a random walk algorithm.

6  
7           **69.**   The software-testing system of claim 66, wherein at least one  
8 algorithm comprises a Chinese Postman algorithm.

9  
10          **70.**   The software-testing system of claim 66, wherein at least one  
11 algorithm comprises a Markov chain algorithm.

12  
13          **71.**   The software-testing system of claim 66, wherein at least one  
14 algorithm comprises a anti-random walk algorithm.

15  
16          **72.**   The software-testing system of claim 66, wherein at least one  
17 algorithm is selected from a group comprising a random walk algorithm, a  
18 Chinese Postman algorithm, a Markov chain algorithm, and a anti-random walk  
19 algorithm.

20  
21          **73.**   A software-testing system comprising:  
22 a software model processor configured to:

23               receive a software model in the form of a state graph that describes  
24 behavior associated with software, the state graph having multiple nodes  
25 that represent state, and links between the nodes that represent actions, and

1           traverse the state graph to provide a sequence of commands for  
2           testing the software;

3           an algorithm set associated with the model processor and comprising  
4           multiple different algorithms; and

5           a graph traverser associated with the model processor and configured to:

6               traverse the state graph using an algorithm from the algorithm set,  
7               the algorithm having a first graph traversal characteristic to produce a  
8               sequence of test commands, and

9               traverse graph with an algorithm from the algorithm set having a  
10              second graph traversal characteristic that is different from the first graph  
11              traversal characteristic to produce a further sequence of test commands.

12  
13       **74.**   The software-testing system of claim 73, wherein the algorithm  
14       having the first graph traversal characteristic is selected from a group of  
15       algorithms comprising: random walk algorithms, random destination algorithms,  
16       and anti-random walk algorithms.

17  
18       **75.**   The software-testing system of claim 74, wherein the algorithm  
19       having the second graph traversal characteristic is selected from a group of  
20       algorithms comprising: random walk algorithms, random destination algorithms,  
21       and anti-random walk algorithms.

22  
23       **76.**   A software-testing system comprising:  
24       means for receiving a software model;  
25



1 means for operating on the software model in a first manner to produce a  
2 sequence of test actions; and

3 means for operating on the software model in different additional manners  
4 to produce additional sequences of test actions.

5  
6 77. The software-testing system of claim 76, wherein said means for  
7 operating comprises multiple different graph traversal algorithms.

8  
9 78. A method of modeling user behavior comprising:  
10 representing software using a model comprising a state graph, the state  
11 graph having multiple nodes individual ones of which represent a state, and links  
12 between the nodes that represent actions;

13 traversing the state graph using an algorithm having a first graph traversal  
14 characteristic to produce a sequence of user actions; and

15 traversing the state graph using an algorithm having a second graph  
16 traversal characteristic that is different from the first graph traversal characteristic  
17 to produce a further sequence of user actions.